图文 021、第3周答疑：本周问题答疑，上周作业点评

1961 人次阅读 2019-07-21 07:00:00

详情 评论

第3周答疑问题汇总

问题

一个面试题：parnew+cms的gc，如何保证只做ygc，jvm参数如何配置？

我的回答:

1)加大分代年龄，比如默认15加到30;

2)修改新生代老年代比例，比如新生代老年代比例改成2:1

3)修改e区和s区比例，比如改成6:2:2 面试官说他们内部服务已经做到fullgc次数为0，只做ygc

想听听老师的意见

答：其实对这个面试题，非常的简单，需要深度结合线上系统的实际运行来看，你如果把本周文章全部看懂了，那么可以轻易回答这个问题。

首先上线系统之后，要借助一些工具（后面会讲）观察每秒钟会新增多少对象在新生代里，然后多长时间触发一次Minor GC，平均每次MInor GC之后会有多少对象存活，Survivor区是否可以放的下。

这里的关键点就是必须让Survivor区放下，而且不能因为动态年龄判定规则直接升入老年代。然后只要Survivor区可以放下，那么下次Minor GC后还是存活这么多对象，依然可以在另外一块Survivor区放下，基本就不会有对象升入老年代里去。

我们的案例其实也是做了类似的优化，几乎消灭了Full GC。包括一些检查参数如何设置

问题：

老师，我们公司的项目是一个数据采集解析的系统。对于每一条采集的数据都会封装成一个对象放到缓存中，然后继续走后续的流程。

假设一个对象（包含原始日志，扩展信息）有1kb大小。一秒一万数据量的话就是一秒将近10mb。然后如果一台给的4g内存给新生代内存。

那么400秒就会触发gc。请问这样算对吗？有什么优化的手段吗？之前开发重来没有考虑过gc问题

答：是的，还得考虑你每条消息需要处理多长时间，如果发生GC，最多可能有多少数据是被引用的，无法回收

问题

我总结了本周所学的知识，得出一套演算过程和公式，请看一下是这样吗？

每台机器可以提供给JVM的最大内存： each_m，比如2核4G机器，可提供JVM最大内存2G 栈占用：stack_m = QPS估值 * 1M * 20倍数，估值30QPS，栈约为600M

新生代以30分钟一次GC计算总内存：30(Monitor GC间隔) * 60 * QPS估值 * 接口内存估值,young_m 所需机器数量，假设等于N

方法区：200M，一般够用，method_m 老年代：500M，一般不大，300M也行，像我们结算服务，100M都够用

old_m 演算公式: JVM最大内存*N = stack_m + young_m + old_m + method_m * N 机器数N，也同时估算出来，是这样吗

答：是这样，总结的非常好

问题

老师，在上面看到一句，gc回收的是软引用，弱应用和虚引用，对吗？

还有我记得jvm是自动去垃圾回收的，程序员没法指定，现在加上弱引用，是去指定回收吗

答：不是的，软引用，弱引用之类的，只不过是指定哪些对象可以回收而已

问题

当初始化静态变量replicaManager的时候，replicaManager成员变量也会初始化，按今天课程的理解，有静态变量引用，所以不会回收。不知道我的理解是否正确。

答：理解正确

学员思考题回答：

思考题：

```
public class Kafka {
    public static ReplicaManager replicaManager = new ReplicaManager();
}
public class ReplicaManager {
    public ReplicaFetcher replicaFetcher = new ReplicaFetcher();
}
```

上述代码下，如果垃圾回收，会回收ReplicaFetcher对象吗？为什么？

学员回答：不会回收，ReplicaFecher对象被replicaFetcher强引用，而ReplicaManager对象又被可作为GC ROOT的replicaManager强引用

所以ReplicaFetcher对象可以向上找到GCROOT，因此不会回收

学员思考题回答：

对于新生代内存清理，首先是栈桢出栈，新生代的引用对象消失，然后就是minoc gc 垃圾回收

问题

老师您好！有个问题想不明白！

比如在方法里有个局部变量A a= new A()，变量a存储在当前方法的栈帧的局部变量表，实例对象存储在堆里！ 在执行到该方法时创建了实例对象

当方法结束后，虚拟机栈都会清空，刚刚执行方法时创建的对象在gc时不会清除？为啥呢，留着还有用吗！那下次再调用这个方法时还会new新的对象啊！想不太明白！

答：此时不会，此时仅仅是对象没人引用而已，要等待垃圾回收的时候给回收掉

问题

打卡。今天内容很简单，不过有一个疑问，就是存活对象在移动的时候内存地址不会发生改变吗？句柄引用的内存地址到底是不是一个固定的值？

答：内存地址会改变

问题

在给新建对象分配内存的时候，被利用的survivor区和Eden区这90%可以看做一个整体，新建对象可以分配到被利用的survivor区，也可以分配到Eden区，只有垃圾回收进行复制的时候才会明显区分出survivor区的作用，这样理解对吗？

答：

对的，理解正确

问题

您好，关于内存大小估算这块，您说可以把之前的计算结果扩大10倍~20倍。

也就是说，每秒钟除了在内存里创建支付订单对象，还会创建其他数十种对象。

为什么是10倍~20倍，为什么您不估算5倍，不估算50倍，每秒中除了在内存里创建支付订单对象，还会创建其他数十种对象这个思路是怎么来的？

望回复，谢谢，如果我们出去面试的话，可能有些面试官会这么问的

答：这是一个经验值，是根据那个支付案例来说的，其实核心含义不是让大家记住这个10~20倍的数字，而是应该你自己根据自己的系统来考虑

不同的系统都是不一样的，你只要把你负责的系统大致看看，就知道他每秒请求过来，会连带创建多少种对象，大致其实就估算出来了

问题

老师讲的思路很清晰，从优缺点分析，有浅入深，能否解答一下：eden区和survivor区 对象是通过什么规则被分配到这两个区的呢？还是说是随机分配

答：优先分配到Eden区，Survivor区是用来放每次GC过后的那些存活对象的

问题

有个地方描述不够严谨吧。比如：接着新对象继续分配在Eden区和另外那块开始被使用的survivor区，然后始终始终保持一块survivor区是空着的。

这个应该没有survivor区是空的吧，之前垃圾回收后不是剩余的存活的对象已经被移动到另一块survivor区了吗？它里面并不是空的啊?是不是这样呢老师

答：假设Eden区和Survivor1区里有对象，一次Minor GC过后，存活对象全部进入Survivor2区域

接着新对象继续在Eden区里分配，Survivor2里放之前Minor GC后存活的对象，然后Survivor1区是空的。

下一次Minor GC过后，剩余存活对象进入Survivor1区里，然后Survivor2区就是空的了。

问题

"接着新的对象继续分配在Eden区和另外那块开始被使用的Survivor区，然后始终保持一块Survivor区是空的"

那么老师，程序在刚刚启动的时候，第一次创建对象时也会分配到Eden区和其中一个Survivror区吗?

还是说只分配到Eden区，此时发现Eden区满了，其他两个survivor区还是空的情况下，触发第一次小GC?

答：只分配到Eden区的，然后第一次Minor GC后会转移存活对象到一块Survivor去

问题

问个问题，为什么老年代不采用复制算法，像新生代那样一个E两个S呢，或者说为什么新生代不采用标记整理算法呢？还有就是标记清除算法会产生内存碎片，那jvm中是在哪里用到了

答：老年代存活对象太多了，如果采用复制算法，每次都挪动可能90%的存活对象，这就不合适了。所以采用先把存活对象挪动到一起紧凑一些，然后回收垃圾对象的方式。

问题

今天的内容比起之前的逻辑性强很多，需要好好思考和梳理一下，感谢老师一步一步带着我们由浅入深。

答：加油，继续坚持

问题

两个问题：

假设MinorGC之前老年代空间担保成功，但是实际不幸的是MinorGC之后老年代放不下而触发了FullGC，之后马上又会伴随一次MinorGC是吗，相当于短时间内进行了两次MinorGC，有这个必要吗？

老年代为什么用标记整理算法？优势在哪里

答：

1、多一次Minor GC没什么，他速度很快

2、老年代存活对象太多了，如果采用复制算法，每次都挪动可能90%的存活对象，这就不合适了。所以采用先把存活对象挪动到一起紧凑一些，然后回收垃圾对象的方式。

问题

大神，默认情况下，新生代占整个堆的3/8，实际情况我们新生代大小占多少合适，我看你在这个支付系统给的是2/3，我们系统在压测的时候，yong gc明显，我把新生代调整为一半了

答：这个新生代设置为多少，其实没有固定的大小，需要自己根据实际系统运行情况来调节

学员的思考题作答：

1、到底什么时候会尝试触发Minor GC?

新生代剩余内存空间放不下新对象，此时需要触发GC。

触发MinorGC情况有:

1- 新生代现有存活对象小于老年代剩余内存 ，即老年空间代足以支撑可能晋升的对象

2- 情况1不成立，查看设置了空间担保且可以担保成功

2、触发Minor GC之前会如何检查老年代大小，涉及哪几个步骤和条件?

1、判断新生代存活是否大于老年代剩余

2、条件1成立且设置空间担保的情况下，判断老年代剩余是否大于之前进入老年代平均存活大小

3、什么时候在Minor GC之前就会提前触发一次Full GC?

新生代现有存活对象>老年代剩余内存情况下，未设置空间担保 或 空间担保失败

4、Full GC的算法是什么?

标记整理算法，速度很慢

5、Minor GC过后可能对应哪几种情况?

放入新对象前进行判断，新对象大小+存活对象是否可以分配在新生代。可以则放入，否则判断是否可以放入老年代。可以则放入，否则 OOM

6、哪些情况下Minor GC后的对象会进入老年代?

新生代放不下，老年代可以放下的情况下

问题

先说一下问题：

1、MinorGC后，存活对象大于survivor区域，小于老年代可用空间，是所有对象都进入老年代，还是一部分

如果所有的对象都进入，是不是会有一些浑水摸鱼？

比如新建的对象，刚好一次MinorGC后还存活，也就是说年龄只有1岁的对象进入老年代，这些对象可能在第二次MinorGC就可以回收，只是正好赶上了这趟车，这样进入老年代没一会就成为垃圾对象了

2、记得之前讲的说设置堆内存，新生代一般是比老年代大的。新生代如果触发MinorGC，表示这新生代满了

一般情况下，新生代满了，对象占用内存肯定是大于老年代可用空间的

所以不明白MinorGC前检查老年代可用空间是否大于新生代所有对象内存，大部分情况下，这个检查的结果都为true呀。难道只是为了文中所说的，规避极端情况吗？

那我直接进行-XX:-HandlePromotionFailure，之后进行FullGC，或MinorGC，如果是极端情况，再抛出OOM，这样不也很好吗。 请老师指点一下

下面是我的今日思考题回答：

1、

Eden区和存对象的Survivor区满的时候触发MinorGC

2、

检查新生对所有对象所占空间是否大于老年代可用空间， 如果小于，进行MinorGC。

如果大于，查看"-XX:-HandlePromotionFailure"是否设置。

如果没设置，进行FullGC。如果设置如果没设置，进行FullGC。

如果设置，判断老年代空间是否大于之前MinorGC后进入老年代对象的平均大小。 如果大于，进行MinorGC。 如果小于，进行FullGC。

3、

(1)新生代对象大小大于老年代空间，且没有设置"-XX:-HandlePromotionFailure"

(2)设置了"-XX:-HandlePromotionFailure"，老年代可用空间小于之前每次MinorGC后进入老年代的平均大小

4、

标记整理算法，老年代对象存活时间较长，赋值算法不太适合，标记-清理算法会产生内存碎片。标记整理可以规避。

5、

(1)小于Survivor区域，进入Survivor区域

(2)大于survivor区域，小于老年代可用空间，进入老年代

(3)大于survivor区域，大于老年代可用空间，进行FullGC，如果FullGC后，老年代可用空间仍小于存活对象，抛出OOM

6、

(1)经过15次(默认，可以设置)MinorGC的

(2)某个年龄的对象大于survivor区域的

答:

1、所有对象都进入老年代

2、不见得，新生代和老年代谁大谁小，看具体场景怎么设置，老年代更大其实很正常，所以一般要设置那个参数，突破第一重检查的限制

问题

是每次Minor GC之前都会去检查是否老年代的大小大于新生代所有内存的大小？还是每次Minor GC之前如果预估到Survivor区域不够了，才会去做这个检查？

如果第一种情况，那么假如Survivor能存的下回收后的对象，每次检查一下老年到貌似不必要。

如果第二种情况，到底是怎么预估的回收后的对象Survivor能放得下？难道遍历一次GCROOT然后计算他们的内存大小。感觉不太靠谱，遍历GC root是垃圾回收器做的事情，发生在“垃圾回收过程中”而不是“垃圾回收之前”

答：每次Minor GC前直接检查新生代全部对象的大小是否小于老年代可用内存大小，不是检查新生代的存活对象的大小，所以这个比较成本是很低的

问题

请教下，我们写好等代码在运行等过程中，对于创建的各种各样的对象，放入新生代的内存顺序一定首先是eden区，然后survivor1区吗？还是也可能是eden区和survivor2区？谢谢老师

答：仅仅放在Eden区里，Survivor是用来放每一次Minor GC后存活的对象的

非常棒的学员思考题回答

到底什么时候会尝试触发Minor GC?

答：当新生代的Eden区和其中一个Survivor区空间不足时。

触发Minor GC之前会如何检查老年代大小，涉及哪几个步骤和条件?

答:

1、先判断新生代中所有对象的大小是否 小于 老年代的可用区域 true 则 触发Minor GC，false则继续进行下面2中的判断

2、如果设置了-XX:HandlePromotionFailure这个参数，那么进入第3步 如果没有设置-XX:HandlePromotionFailure参数，那么触发Full GC

3、判断Minor GC历次进入老年代的平均大小是否 小于 老年代的可用区域 true 则触发Minor GC，false则会触发Full GC 什么时候在Minor GC之前就会提前触发一次Full GC?

答：当判断 新生代历次进入老年代对象的平均大小 大于 老年代的可用区域就会触发一次Full GC，让老年代腾出一些空间，腾出空间后再进行Minor GC。

FullGC的算法是什么?

答：标记整理算法。

Minor GC过后可能对应哪几种情况?

答：

情况1：Minor GC前先判断：存活的对象所占的内存空间 < Survivor区域内存空间的大小，那么存活的对象进入Survivor区。

情况2：Minor GC前先判断：Survivor区域内存空间的大小 < 存活的对象所占的内存空间 < 老年代的可用空间大小。那么存活的对象，直接进入老年代。

情况3：Minor GC前先判断： (存活的对象所占的内存空间 > Survivor区域内存空间的大小) && (存活的对象所占的内存空间 > 老年代的可用空间大小)。那么会触发Full GC，老年代腾出空间后，再进行Minor GC。如果腾出空间后还不能存放存活的对象，那么会导致OOM即堆内存空间不足、堆内存溢出。

哪些情况下Minor GC后的对象会进入老年代?

答：

1、 Survivor区 < 存活对象占用的空间 && 老年代可用区域 < 存活对象占用的空间

2、经过XX:MaxTenuringThreshold次M

问题

老师，诚心求教，希望能帮忙解释清楚，反复看了几次，确实文章开头说的也是我们写好的代码在运行过程中，会不断的创建各种各样的对象

这些对象会优先存放在新生代的Eden和survivor1区，接着假如新生代的Eden和s1都快满了，就发生minor gc，存活对象移动到s2.

然后接着就会使用Eden和s2区来分配新的对象

那文章下面为什么又提到系统继续运行，继续在Eden区里分配各种对象？难道不该是系统继续运行，继续在Eden区和s2区里分配各种对象，发生minor gc后，再移动到s1吗？这里的描述前后理解困惑

答：同学，你可能理解错一点了，对象都是在Eden区分配的，然后一次Minor GC过后，存活对象转移到S1区，此时Eden区就清空了，对吧？

然后接着系统继续运行，对象继续分配在Eden区，下一次Minor GC过后，看看Eden + S1里的存活对象有哪些，都转移到S2区。

接着系统继续运行，对象继续分配在Eden区，下一次Minor GC过后，看看Eden + S2里的存活对象有哪些，都转移到S1区。S1和S2就是用来放存活对象的，Eden区用来让系统分配新的对象进去。

问题

非常感激老师的耐心解答，那是不是可以理解为minor gc触发条件是Eden区放不下新的对象为触发条件？

然后追问一个关于full gc，是不是存在minor gc检查历次的放入老年代的对象大小小于本次实际要挪移的对象大小情况，从而才会说有minor gc发生后，再次full gc再minor gc的情况？感谢，感谢

答：

1、对的

2、不对，是历次进入老年代的对象大小小于当前老年代的内存大小，才会触发minor gc；否则先是full gc给老年代腾空间，再minor gc

问题

忍不住再追加一个确认，会不会存在minor gc前检查历次进入老年代的对象均值是10M，然后当前老年代剩余50M，明显大于历次的10M这个值，于是触发minor gc正常执行，但执行后发现本次实际要放入老年代的对象大小大于均值且大于50M的情况，从而再来一次fullgc和minor gc这样的情况？感恩分享

答：会的，如果是这种情况，就是会先执行minor gc，然后发现老年代内存不足，触发full gc

问题

老师，每次Minor GC后剩余的是200MB是根据2 8法则推测的吗，我们平时做优化假设也是按照这个法则来吗？那如果我想知道实际中真正剩余多少内存的对象有什么办法吗？

答：不是的，这个数字是根据线上生产日志判断出来的，后续会教大家看gc日志，很清晰可以看到每分钟minor gc的时候多少对象进入老年代

问题

打卡。今天这个案例很好的结合估算内存压力大小和新老代回收机制进行了产线问题排查，这个思路在以后的工作中也很重要，学习了。

答：是的，建议直接按照这个案例背景从头到尾画一下整个内存分析的全流程和优化过程

问题

老师，评估每次任务或者每秒系统产生的数据大小是不是有专门的工具可以结合起来看。

答：是的，后续会带着动手操作一些工具的使用，这里就是先讲原理，结合案例说明原理

问题

每个计算计算任务1万条数据需要计算10秒钟，假设此时计算出80个计算任务都执行成功了。。。卡在这里了

后来想明白了：假设1分钟100个计算任务，每个计算任务是10秒钟，在60秒到来的那一刻，只有50-60秒开始执行的计算任务是没有执行完毕的，占比1/6,100 / 6 = 16.66 。 所以可以大致估算还有20个计算任务还在计算中。

答：是的，线上生产系统大致就是这个比例

问题

关于空间分配担保有一点疑问 当老年代的可用内存空间小于新生代所有对象的总大小时, 无论有没有空间分配担保, 最坏的情况都是一次minor GC + Full Gc

那为什么还需要这个分配担保?? 直接minor GC, 之后再进行那三种可能的判断不就行了, HandlePromotionFail的意义何在?

答：不是的，如果说没有开启一个检查，此时可能提前Full GC，那么这样就太频繁了；但是如果经过文章里说的检查机制，发现不需要Full GC那么就直接Minor GC；差别在于不需要频繁Full GC

问题

按照每一次Minor GC之前JVM都会检测老年代空间是否大于新生代对象的总大小。一般上线的环境下，控制Minor GC的频率后，如果资源是不是是尽量为老年代分配更大的空间。如果不行，就需要设置参数HandlePromationFail。

这样理解可以吗？老师

答：理解正确的，但是其实具体一个系统要如何分配Eden、Survivor、老年代的内存，其实是很有讲究的，主要得看系统运行的特点